OPTIMIZATION OF NUMBER OF TRANSCEIVERS USED IN A SWITCH

FIELD OF THE INVENTION

The present invention is related to transferring fragments between port cards and fabrics. More specifically, the present invention is related to transferring fragments between port cards and fabrics with gigabit transceivers.

BACKGROUND OF THE INVENTION

The switch originally was designed to have high-speed HSTL point-to-point connections between port cards and fabrics running at 250 MHz. This was proven to be too fast and signal integrity could not be guaranteed due to the long trace length that exist. A more robust method had to be devised. The new method uses a network of gigabit transceivers between port cards and fabrics. The gigabit network is comprised of gigabit transmitters and receivers that communicate with each other at up to 1.3GHz. The interfaces to each transceiver are an 8-bit data bus running at 125 MHz. Since the frequency of operation was reduced by half, the bus widths had to be multiplied by two. This new method provides to be more robust and has better noise rejection since the gigabit transceivers use differential pairs to communicate with each other. Since these gigabit transceivers are replacing point-to-point buses and some buses are larger than 8 bits wide, multiple gigabit transceivers need to be used to make up these buses. The initial total number of gigabit transceivers required was as follows: 22 transmitters per Striper, 40 receivers per Aggregator, 52 transmitters per Separator and 28 receivers per Unstriper. For a 480G switch, this would total to about (48 port cards)(4

Stripers)(22) + (13 fabrics)(9 Aggregators)(40) + (13 fabrics)(9 Separators)(52) + (48 port cards)(4 Unstripers)(28) = 4224 + 4680 + 6084 + 376 = 20364. This large number of gigabit transceivers proved to consume too much power and take up too much boards space. At an average of 0.29 Watts of power dissipation per transceiver, the total power dissipation would have been 5,905 Watts (20364)(.029).

The present invention using the gigabit transceivers solves the following problems:

1. Reduces the total number of gigabit transceivers required.
2. Reduces total power requirements of whole system.
3. Reduces board space requirements.
4. Handles un-synchronized updates to gigabit network.
5. Minimizes I/O requirements on backplane connectors.
6. Reduces PCB trace routing complexity.
7. Maximizes symmetry between ASICs which will simplify unique mux/demux structures required.

This optimization solves all the above problems by intelligently allocating gigabit transceivers to be re-used in modes where they can be re-used. This technique allows changes in the configuration of the switch (40G, 80G, 120G, etc.) To alter the gigabit transmitter and receiver assignment to be un-synchronized as long as the switch configuration is changed up or down by one step in the configuration.

A switch which stripes data onto multiple fabrics and sends parity data to another fabric has been described in U.S.

patent application serial number 09/333,450, incorporated by reference herein. See also U.S. patent application serial number 09/293,563 which describes a wide memory TDM switching system, incorporated by reference herein.

SUMMARY OF THE INVENTION

The present invention pertains to a switch. The switch comprises a plurality of port cards. The switch comprises a network connected to the port cards. Preferably, the network is a gigabit network. The gigabit network has transmitters and receivers that communicate with each other and have assignments between each other. Preferably, the transmitters and receivers are gigabit transmitters and gigabit receivers. The switch comprises a plurality of fabrics. Each fabric is connected to each port card through the gigabit network to send and receive fragments of packets to or from the port cards, the port cards, fabrics and network having a plurality of modes of operation. The switch comprises a control mechanism connected to the gigabit transmitters and gigabit receivers which changes the assignments according to the mode. The control mechanism changes the mode and reuses the gigabit transmitters and gigabit receivers where they can be reused.

The present invention pertains to a method for switching fragments of packets. The method comprises the steps of assigning assignments between transmitters and receivers of a network. Preferably, the network is a gigabit network. Preferably, the transmitters and receivers are gigabit transmitters and gigabit receivers. Next there is the step of changing a mode of the

fabrics, port cards and networks. Then there is the step of changing the assignments of the transmitters and receivers according to the mode and reusing the gigabit transmitters and gigabit receivers where they can be reused. Then there is the step of transferring the fragments of packets between fabrics and port cards with the gigabit transmitters and gigabit receivers of the gigabit network.

BRIEF DESCRIPTION OF THE DRAWINGS

In the accompanying drawings, the preferred embodiment of the invention and preferred methods of practicing the invention are illustrated in which:

Figure 1 is a schematic representation of packet striping in the switch of the present invention.

Figure 2 is a schematic representation of an OC 48 port card.

Figure 3 is a schematic representation of a concatenated network blade.

Figure 4 is a schematic representation regarding the connectivity of the fabric ASICs.

Figure 5 is a schematic representation of sync pulse distribution.

Figure 6 is a schematic representation regarding the relationship between transmit and receive sequence counters for the separator and unstriper, respectively.

Figure 7 is a schematic representation of a switch of the present invention.

DETAILED DESCRIPTION

Referring now to the drawings wherein like reference numerals refer to similar or identical parts throughout the several views, and more specifically to figure 7 thereof, there is shown a switch 10. The switch 10 comprises at least one port card 12. The switch 10 comprises a network 14 connected to the port card 12. Preferably, the network is a gigabit network. The gigabit network 14 has transmitters 16 and receivers 18 that communicate with each other and have assignments between each other. Preferably, the transmitters and receivers are gigabit transmitters and gigabit receivers. The switch 10 comprises at least one fabric 20. Each fabric 20 is connected to each port card 12 through the gigabit network 14 to send and receive fragments of packets to or from the port cards 12, the port cards 12, fabrics 20 and network 14 having a plurality of modes of operation. The switch 10 comprises a control mechanism 22 connected to the gigabit transmitters 16 and gigabit receivers 18 which changes the assignments according to the mode. The control mechanism 22 changes the mode and reuses the gigabit transmitters 16 and gigabit receivers 18 where they can be reused.

Preferably, the network 14 includes a mux structure 24 that makes the assignments between transmitters 16 and receivers 18. The modes preferably are 40G, 80G, 120G, 240G slow, 240G fast or 480G. Preferably, the network 14 can support two modes simultaneously as combined modes. The combined modes preferably are 40G/80G, 80G/120G, 120G/160G, 160G/240G fast, 240G fast/240G slow, or 240G slow/480G.

Preferably, each port card 12 includes a striper 26 and an unstriper 28. Each fabric 20 preferably includes a separator 30 and an aggregator 32. Preferably, the assignments between transmitters 16 and receivers 18 communicate with each other at up to 1.3 GHz. Each transmitter 16 preferably takes in 8 bits of data and 2 bits of control and serially transmits the bits of data and control to the associated receiver. Preferably, each receiver recovers clock and data it receives by using an 8B/10B decoding protocol and provides 8 bits of data and 3 bits of control.

The present invention pertains to a method for switching fragments of packets. The method comprises the steps of assigning assignments between transmitters 16 and receivers 18 of a network 14. Preferably, the network is a gigabit network. Preferably, the transmitters and receivers are gigabit transmitters and gigabit receivers. Next there is the step of changing a mode of the fabrics 20, port cards 12 and networks 14. Then there is the step of changing the assignments of the transmitters 16 and receivers 18 according to the mode and reusing the gigabit transmitters 16 and gigabit receivers 18 where they can be reused. Then there is the step of transferring the fragments of packets between fabrics 20

and port cards 12 with the gigabit transmitters 16 and gigabit receivers 18 of the gigabit network 14.

Preferably, the changing of the assignments step includes the step of rearranging the mux structure 24 of the network 14 that makes the assignments between the receivers 18 and transmitters 16. The changing the mode step preferably includes the step of changing the mode between 40G and 80G, or 80G and 120G, or 120G and 160G or 160G and 240G slow or 240G slow and 240G fast. Preferably, the assigning step includes the step of assigning assignments between the receivers 18 and transmitters 16 to support two modes simultaneously as combined modes. The transferring step preferably includes the step of transferring with the transmitter 8 bits of data and 2 bits of control serially through a mux structure 24 to the receiver assigned to the transmitter.

In the operation of the invention, the switch uses a network 14 of gigabit transceivers that make the connections from port cards 12 to fabrics 20 across the backplane. The gigabit network 14 is comprised of gigabit transmitters 16 and receivers 18 that communicate with each other at up to 1.3 GHz by using differential pairs. Each gigabit transmitter 16 takes in 8 bits of data and 2 bits of control and serially transmits this data to a gigabit receiver that recovers clock and data and by using an 8B/10B decoding protocol, provides 8 bits of data and 3 bits of gigabit receiver that recovers clock and data and by using an 8B/10B decoding protocol, provides 8 bits of data and 3 bits of control. Between the transmitters 16 and receivers 18 exists a mux structure 24 that makes the assignment of transmitters 16 to receivers 18. The mux structure 24 can be configured by software

control to re-assign transceivers based on the mode of the switch 10. The mux structure 24 utilizes built-in 2-to-1 muxes in each transceiver. The mux structure 24 could not be built entirely with 2-to-1 muxes since in some cases 3-to-1 muxes were required. Therefore, some additional external 2-to-1 muxes are required. The seven possible configurations are as follows 40G, 80G, 120G, 160G, 240G slow, 240G fast, 480G. Since updates to the gigabit network 14 cannot be synchronized across the whole switch 10, it was decided to combine the switch 10 configurations in pairs so that at any particular time, the gigabit network 14 would support two configurations simultaneously. This can be done because pairs of configuration modes have transceivers that are common to both and can be shared. However, the switch 10 configuration is restricted to move up or down in capacity by one mode at a time. The combined modes that are supported as follows: 40G/80G, 80G/120G, 120G/160G, 160G/240G fast, 240G fast/240G slow, 240G slow/480G. 240G fast and slow modes are described later.

The minimum requirement on the number of gigabit transceivers can be obtained by looking at the minimum number of independent channels that need to be supported by each ASIC. For the Striper 26 and Unstriper 28, 13 independent sources and destinations exist respectively, that need their own clock source in 480G mode. The Striper 26 needs to source 2 routeward bits, 2 data bits and a back-pressure bit for a total of 5 bits per transmitter used. The Separator 30 needs to source 4 routeword bits and 4 data bits for a combined total of 8 bits per transmitter. Each transceiver can handle 8 bits of data so one transceiver per channel in 480G mode is sufficient. This limits the reduction of gigabit transceivers to at least 13 per striper

and unstriper since each gigabit transceiver provides only one clock source. A similar requirement holds true for the Aggregator 32 and Separator 30. The Aggregator 32 and Separator 30 require 24 independent channels in 480G mode and therefore require a minimum of 24 gigabit transceivers each. After some analysis, it was shown that the original speedup of 2 on the egress of the switch was compromised in some modes. The speedup for 240G mode was reduced to 1 and was not acceptable. To have a speedup of 2 in 240G mode, one additional transceiver was required for the Unstriper 28. To make things symmetrical, an additional transmitter 16 was added to the Striper 26. 240G mode was split into two modes. 240G fast mode has a speedup of 2 and 240G slow mode has a speedup of 1. 240G slow mode needs to also be used since no common mapping of transceivers could be found from 240G fast mode to 480G mode. This is true since both modes use all 14 transceivers but have completely different mappings. This can be seen in the Appendix "BFS Backplane Optimization Worksheet". This shows how the gigabit transceiver assignments were derived. The original map was done for the Unstriper 28 and from this, it can be seen that in 240G fast mode, all 14 transceivers were required. All other modes were assigned based on the 240G fast mode and unused transceivers were re-used in modes that required them. The Striper 26 map is a subset of the Unstriper 28 map and both Aggregator 32 maps and the Separator 30 5-8 map are derived from the Separator 30 1-4 map. In the Appendix, the "BFS ASIC Backplane Connection Map" shows the final gigabit transceiver assignments. This is a summary of the mappings derived earlier. The document in the Appendix "Gigabit Signals", shows how the number of data bits per bus for each mode affect the number of gigabit transceivers used.

After this optimization, the total gigabit transceiver requirement is as follows: 14 transmitters per Striper, 24 receivers 18 per Aggregator, 24 transmitters per Separator and 14 receivers per Unstriper 28. For a 480G switch, this would total to about (48 port cards)(4 Stripers)(14) + (13 fabrics)(9 Aggregators)(24) + (13 fabrics)(9 Separators)(24) + (48 port cards)(4 Unstripers)(14) = 2688 + 2808 + 2808 + 2688 = 10992. This is a 46% reduction in the number of transceivers required. At an average of 0.29 Watts of power dissipation per transceiver, the total power dissipation will be 3,187 Watts (10992)(0.29).

The switch uses RAID techniques to increase overall switch bandwidth while minimizing individual fabric bandwidth. In the switch architecture, all data is distributed evenly across all fabrics so the switch adds bandwidth by adding fabrics and the fabric need not increase its bandwidth capacity as the switch increases bandwidth capacity.

Each fabric provides 40G of switching bandwidth and the system supports 1, 2, 3, 4, 6, or 12 fabrics, exclusive of the redundant/spare fabric. In other words, the switch can be a 40G, 80G, 120G, 160G, 240G, or 480G switch depending on how many fabrics are installed.

A portcard provides 10G of port bandwidth. For every 4 portcards, there needs to be 1 fabric. The switch architecture does not support arbitrary installations of portcards and fabrics.

The fabric ASICs support both cells and packets. As a whole, the switch takes a "receiver make right" approach where the

egress path on ATM blades must segment frames to cells and the egress path on frame blades must perform reassembly of cells into packets.

There are currently eight switch ASICs that are used in the switch:

1. Striper - The Striper resides on the portcard and SCP-IM. It formats the data into a 12 bit data stream, appends a checkword, splits the data stream across the N, non-spare fabrics in the system, generates a parity stripe of width equal to the stripes going to the other fabric, and sends the N+1 data streams out to the backplane.

2. Unstriper - The Unstriper is the other portcard ASIC in the the switch architecture. It receives data stripes from all the fabrics in the system. It then reconstructs the original data stream using the checkword and parity stripe to perform error detection and correction.

3. Aggregator - The Aggregator takes the data streams and routewords from the Stripers and multiplexes them into a single input stream to the Memory Controller.

4. Memory Controller - The Memory controller implements the queueing and dequeueing mechanisms of the switch. This includes the proprietary wide

memory interface to achieve the simultaneous en-/de-queueing of multiple cells of data per clock cycle. The dequeueing side of the Memory Controller runs at 80Gbps compared to 40Gbps in order to make the bulk of the queueing and shaping of connections occur on the portcards.

5. Separator - The Separator implements the inverse operation of the Aggregator. The data stream from the Memory Controller is demultiplexed into multiple streams of data and forwarded to the appropriate Unstriper ASIC. Included in the interface to the Unstriper is a queue and flow control handshaking.

There are 3 different views one can take of the connections between the fabric: physical, logical, and "active." Physically, the connections between the portcards and the fabrics are all gigabit speed differential pair serial links. This is strictly an implementation issue to reduce the number of signals going over the backplane. The "active" perspective looks at a single switch configuration, or it may be thought of as a snapshot of how data is being processed at a given moment. The interface between the fabric ASIC on the portcards and the fabrics is effectively 12 bits wide. Those 12 bits are evenly distributed ("striped") across 1, 2, 3, 4, 6, or 12 fabrics based on how the fabric ASICs are configured. The "active" perspective refers to the number of bits being processed by each fabric in the current configuration which is exactly 12 divided by the number of fabrics.

The logical perspective can be viewed as the union or max function of all the possible active configurations. Fabric slot #1 can, depending on configuration, be processing 12, 6, 4, 3, 2, or 1 bits of the data from a single Striper and is therefore drawn with a 12 bit bus. In contrast, fabric slot #3 can only be used to process 4, 3, 2, or 1 bits from a single Striper and is therefore drawn with a 4 bit bus.

Unlike previous switches, the switch really doesn't have a concept of a software controllable fabric redundancy mode. The fabric ASICs implement N+1 redundancy without any intervention as long as the spare fabric is installed.

As far as what does it provide; N+1 redundancy means that the hardware will automatically detect and correct a single failure without the loss of any data.

The way the redundancy works is fairly simple, but to make it even simpler to understand a specific case of a 120G switch is used which has 3 fabrics (A, B, and C) plus a spare (S). The Striper takes the 12 bit bus and first generates a checkword which gets appended to the data unit (cell or frame). The data unit and checkword are then split into a 4-bit-per-clock-cycle data stripe for each of the A, B, and C fabrics ($A_3A_2A_1A_0$, $B_3B_2B_1B_0$, and $C_3C_2C_1C_0$). These stripes are then used to produce the stripe for the spare fabric $S_3S_2S_1S_0$ where $S_n = A_n$ XOR B_n XOR C_n and these 4 stripes are sent to their corresponding fabrics. On the other side of the fabrics, the Unstriper receives 4 4-bit stripes from A, B, C, and S. All possible combinations of 3 fabrics (ABC, ABS, ASC, and SBC) are then used to reconstruct a "tentative" 12-bit data stream. A

checkword is then calculated for each of the 4 tentative streams and the calculated checkword compared to the checkword at the end of the data unit. If no error occurred in transit, then all 4 streams will have checkword matches and the ABC stream will be forwarded to the Unstriper output. If a (single) error occurred, only one checkword match will exist and the stream with the match will be forwarded off chip and the Unstriper will identify the faulty fabric stripe.

For different switch configurations, i.e. 1, 2, 4, 6, or 12 fabrics, the algorithm is the same but the stripe width changes.

If 2 fabrics fail, all data running through the switch will almost certainly be corrupted.

The fabric slots are numbered and must be populated in ascending order. Also, the spare fabric is a specific slot so populating fabric slots 1, 2, 3, and 4 is different than populating fabric slots 1, 2, 3, and the spare. The former is a 160G switch without redundancy and the latter is 120G with redundancy.

Firstly, the ASICs are constructed and the backplane connected such that the use of a certain portcard slots requires there to be at least a certain minimum number of fabrics installed, not including the spare. This relationship is shown in Table 0.

In addition, the APS redundancy within the switch is limited to specifically paired portcards. Portcards 1 and 2 are paired, 3 and 4 are paired, and so on through portcards 47 and 48.

This means that if APS redundancy is required, the paired slots must be populated together.

To give a simple example, take a configuration with 2 portcards and only 1 fabric. If the user does not want to use APS redundancy, then the 2 portcards can be installed in any two of portcard slots 1 through 4. If APS redundancy is desired, then the two portcards must be installed either in slots 1 and 2 or slots 3 and 4.

Portcard Slot	**Minimum # of Fabrics**
1-4	1
5-8	2
9-12	3
13-16	4
17-24	6
25-48	12

Table 0: Fabric Requirements for Portcard Slot Usage

To add capacity, add the new fabric(s), wait for the switch to recognize the change and reconfigure the system to stripe across the new number of fabrics. Install the new portcards.

Note that it is not technically necessary to have the full 4 portcards per fabric. The switch will work properly with 3 fabrics installed and a single portcard in slot 12. This isn't cost efficient but it will work.

To remove capacity, reverse the adding capacity procedure.

If the switch is oversubscribed, i.e. install 8 portcards and only one fabric.

It should only come about as the result of improperly upgrading the switch or a system failure of some sort. The reality is that one of two things will occur, depending on how this situation arises. If the switch is configured as a 40G switch and the portcards are added before the fabric, then the 5^{th} through 8^{th} portcards will be dead. If the switch is configured as 80G non-redundant switch and the second fabric fails or is removed then all data through the switch will be corrupted (assuming the spare fabric is not installed). And just to be complete, if 8 portcards were installed in an 80G redundant switch and the second fabric failed or was removed, then the switch would continue to operate normally with the spare covering for the failed/removed fabric.

Figure 1 shows packet striping in the switch.

The chipset supports ATM and POS port cards in both OC48 and OC192c configurations. OC48 port cards interface to the switching fabrics with four separate OC48 flows. OC192 port cards logically combine the 4 channels into a 10G stream. The ingress side of a port card does not perform traffic conversions for traffic changing between ATM cells and packets. Whichever form of traffic is received is sent to the switch fabrics. The switch fabrics will mix packets and cells and then dequeue a mix of packets and cells to the egress side of a port card.

The egress side of the port is responsible for converting the traffic to the appropriate format for the output port. This

convention is referred to in the context of the switch as "receiver makes right". A cell blade is responsible for segmentation of packets and a cell blade is responsible for reassembly of cells into packets. To support fabric speed-up, the egress side of the port card supports a link bandwidth equal to twice the inbound side of the port card.

The block diagram for a Poseidon-based ATM port card is shown as in Figure 2. Each 2.5G channel consists of 4 ASICs: Inbound TM and striper ASIC at the inbound side and unstriper ASIC and outbound TM ASIC at the outbound side.

At the inbound side, OC-48c or 4 OC-12c interfaces are aggregated. Each vortex sends a 2.5G cell stream into a dedicated striper ASIC (using the BIB bus, as described below). The striper converts the supplied routeword into two pieces. A portion of the routeword is passed to the fabric to determine the output port(s) for the cell. The entire routeword is also passed on the data portion of the bus as a routeword for use by the outbound memory controller. The first routeword is termed the "fabric routeword". The routeword for the outbound memory controller is the "egress routeword".

At the outbound side, the unstriper ASIC in each channel takes traffic from each of the port cards, error checks and correct the data and then sends correct packets out on its output bus. The unstriper uses the data from the spare fabric and the checksum inserted by the striper to detect and correct data corruption.

Figure 2 shows an OC48 Port Card.

The OC192 port card supports a single 10G stream to the fabric and between a 10G and 20G egress stream. This board also uses 4 stripers and 4 unstriper, but the 4 chips operate in parallel on a wider data bus. The data sent to each fabric is identical for both OC48 and OC192 ports so data can flow between the port types without needing special conversion functions.

Figure 3 shows a 10G concatenated network blade.

Each 40G switch fabric enqueues up to 40Gbps cells/frames and dequeue them at 80Gbps. This 2X speed-up reduces the amount of traffic buffered at the fabric and lets the outbound ASIC digest bursts of traffic well above line rate. A switch fabric consists of three kinds of ASICs: aggregators, memory controllers, and separators. Nine aggregator ASICs receive 40Gbps of traffic from up to 48 network blades and the control port. The aggregator ASICs combine the fabric route word and payload into a single data stream and TDM between its sources and places the resulting data on a wide output bus. An additional control bus (destid) is used to control how the memory controllers enqueue the data. The data stream from each aggregator ASIC then bit sliced into 12 memory controllers.

The memory controller receives up to 16 cells/frames every clock cycle. Each of 12 ASICs stores 1/12 of the aggregated data streams. It then stores the incoming data based on control information received on the destid bus. Storage of data is simplified in the memory controller to be relatively unaware of packet boundaries (cache line concept). All 12 ASICs dequeue the stored cells simultaneously at aggregated speed of 80Gbps.

Nine separator ASICs perform the reverse function of the aggregator ASICs. Each separator receives data from all 12 memory controllers and decodes the routewords embedded in the data streams by the aggregator to find packet boundaries. Each separator ASIC then sends the data to up to 24 different unstripers depending on the exact destination indicated by the memory controller as data was being passed to the separator.

The dequeue process is back-pressure driven. If back-pressure is applied to the unstriper, that back-pressure is communicated back to the separator. The separator and memory controllers also have a back-pressure mechanism which controls when a memory controller can dequeue traffic to an output port.

In order to support OC48 and OC192 efficiently in the chipset, the 4 OC48 ports from one port card are always routed to the same aggregator and from the same separator (the port connections for the aggregator & Sep are always symmetric.). The table below shows the port connections for the aggregator & sep on each fabric for the switch configurations. Since each aggregator is accepting traffic from 10G of ports, the addition of 40G of switch capacity only adds ports to 4 aggregators. This leads to a differing port connection pattern for the first four aggregators from the second 4 (and also the corresponding separators).

TABLE 2: Agg/Sep port connections

Switch Size	Agg 1	Agg 2	Agg 3	Agg 4	Agg 5	Agg 6	Agg 7	Agg 8
40	1,2,3,4	5,6,7,8	9,10,11,12	13,14,15, 16				
80	1,2,3,4	5,6,7,8	9,10,11,12	13,14,15, 16	17,18,19, 20	21,22,23, 24	25,26,27, 28	29,30,31, 32
120	1,2,3,4	5,6,7,8	9,10,11,12,	13,14,15, 16,	17,18,19, 20	21,22,23, 24	25,26,27, 28	29,30,31, 32
	33,34,35, 36	37,38,39, 40	41,42,43, 44	45,46,47, 48				
160	1,2,3,4	5,6,7,8	9,10,11,12,	13,14,15, 16,	17,18,19, 20,	21,22,23, 24,	25,26,27, 28,	29,30,31, 32,

33,34,35, 36 37,38,39, 40 41,42,43, 44 45,46,47, 48 49,50,51, 52 53,54,55, 56 57,58,59, 60 61,62,63, 64

Figure 4 shows the connectivity of the fabric ASICs.

The external interfaces of the switches are the Input Bus (BIB) between the striper ASIC and the ingress blade ASIC such as Vortex and the Output Bus (BOB) between the unstriper ASIC and the egress blade ASIC such as Trident.

The unstriper ASIC sends data to the egress port via Output Bus (BOB) (also known as DOUT_UN_bl_ch bus), which is a 64 (or 256) bit data bus that can support either cell or packet. It consists of the following signals:

This bus can either operate as 4 separate 32 bit output buses (4xOC48c) or a single 128 bit wide data bus with a common set of control lines from all Unstripers. This bus supports either cells or packets based on software configuration of the unstriper chip.

The Synchronizer has two main purposes. The first purpose is to maintain logical cell/packet or datagram ordering across all fabrics. On the fabric ingress interface, datagrams arriving at more than one fabric from one port cards's channels need to be processed in the same order across all fabrics. The Synchronizer's second purpose is to have a port cards's egress channel re-assemble all segments or stripes of a datagram that belong together even though the datagram segments are being sent from more than one fabric and can arrive at the blade's egress inputs at different times. This mechanism needs to be maintained in

a system that will have different net delays and varying amounts of clock drift between blades and fabrics.

The switch uses a system of a synchronized windows where start information is transmit around the system. Each transmitter and receiver can look at relative clock counts from the last resynch indication to synchronize data from multiple sources. The receiver will delay the receipt of data which is the first clock cycle of data in a synch period until a programmable delay after it receives the global synch indication. At this point, all data is considered to have been received simultaneously and fixed ordering is applied. Even though the delays for packet 0 and cell 0 caused them to be seen at the receivers in different orders due to delays through the box, the resulting ordering of both streams at receive time = 1 is the same, Packet 0, Cell 0 based on the physical bus from which they were received.

Multiple cells or packets can be sent in one counter tick. All destinations will order all cells from the first interface before moving onto the second interface and so on. This cell synchronization technique is used on all cell interfaces. Differing resolutions are required on some interfaces.

The Synchronizer consists of two main blocks, mainly, the transmitter and receiver. The transmitter block will reside in the Striper and Separator ASICs and the receiver block will reside in the Aggregator and Unstriper ASICs. The receiver in the Aggregator will handle up to 24(6 port cards x 4 channels) input lanes. The receiver in the Unstriper will handle up to 13(12 fabrics + 1 parity fabric) input lanes.

When a sync pulse is received, the transmitter first calculates the number of clock cycles it is fast (denoted as N clocks).

The transmit synchronizer will interrupt the output stream and transmit N K characters indicating it is locking down. At the end of the lockdown sequence, the transmitter transmits a K character indicating that valid data will start on the next clock cycle. This next cycle valid indication is used by the receivers to synchronize traffic from all sources.

At the next end of transfer, the transmitter will then insert at least one idle on the interface. These idles allow the 10 bit decoders to correctly resynchronize to the 10 bit serial code window if they fall out of synch.

The receive synchronizer receives the global synch pulse and delays the synch pulse by a programmed number (which is programmed based on the maximum amount of transport delay a physical box can have). After delaying the synch pulse, the receiver will then consider the clock cycle immediately after the synch character to be eligible to be received. Data is then received every clock cycle until the next synch character is seen on the input stream. This data is not considered to be eligible for receipt until the delayed global synch pulse is seen.

Since transmitters and receivers will be on different physical boards and clocked by different oscillators, clock speed differences will exist between them. To bound the number of clock cycles between different transmitters and receivers, a global sync

pulse is used at the system level to resynchronize all sequence counters. Each chip is programmed to ensure that under all valid clock skews, each transmitter and receiver will think that it is fast by at least one clock cycle. Each chip then waits for the appropriate number of clock cycles they are into their current sync_pulse_window. This ensure that all sources run N* sync_pulse_window valid clock cycles between synch pulses.

As an example, the synch pulse window could be programmed to 100 clocks, and the synch pulses sent out at a nominal rate of a synch pulse every 10,000 clocks. Based on a worst case drifts for both the synch pulse transmitter clocks and the synch pulse receiver clocks, there may actually be 9,995 to 10,005 clocks at the receiver for 10,000 clocks on the synch pulse transmitter. In this case, the synch pulse transmitter would be programmed to send out synch pulses every 10,006 clock cycles. The 10,006 clocks guarantees that all receivers must be in their next window. A receiver with a fast clock may have actually seen 10,012 clocks if the synch pulse transmitter has a slow clock. Since the synch pulse was received 12 clock cycles into the synch pulse window, the chip would delay for 12 clock cycles. Another receiver could seen 10,006 clocks and lock down for 6 clock cycles at the end of the synch pulse window. In both cases, each source ran 10,100 clock cycles.

When a port card or fabric is not present or has just been inserted and either of them is supposed to be driving the inputs of a receive synchronizer, the writing of data to the particular input FIFO will be inhibited since the input clock will not be present or unstable and the status of the data lines will be

unknown. When the port card or fabric is inserted, software must come in and enable the input to the byte lane to allow data from that source to be enabled. Writes to the input FIFO will be enabled. It is assumed that, the enable signal will be asserted after the data, routeword and clock from the port card or fabric are stable.

At a system level, there will be a primary and secondary sync pulse transmitter residing on two separate fabrics. There will also be a sync pulse receiver on each fabric and blade. This can be seen in Figure 5. A primary sync pulse transmitters will be a free-running sync pulse generator and a secondary sync pulse transmitter will synchronize its sync pulse to the primary. The sync pulse receivers will receive both primary and secondary sync pulses and based on an error checking algorithm, will select the correct sync pulse to forward on to the ASICs residing on that board. The sync pulse receiver will guarantee that a sync pulse is only forwarded to the rest of the board if the sync pulse from the sync pulse transmitters falls within its own sequence "0" count. For example, the sync pulse receiver and an Unstriper ASIC will both reside on the same Blade. The sync pulse receiver and the receive synchronizer in the Unstriper will be clocked from the same crystal oscillator, so no clock drift should be present between the clocks used to increment the internal sequence counters. The receive synchronizer will require that the sync pulse it receives will always reside in the "0" count window.

If the sync pulse receiver determines that the primary sync pulse transmitter is out of sync, it will switch over to the secondary sync pulse transmitter source. The secondary sync pulse

transmitter will also determine that the primary sync pulse transmitter is out of sync and will start generating its own sync pulse independently of the primary sync pulse transmitter. This is the secondary sync pulse transmitter's primary mode of operation. If the sync pulse receiver determines that the primary sync pulse transmitter has become in sync once again, it will switch to the primary side. The secondary sync pulse transmitter will also determine that the primary sync pulse transmitter has become in sync once again and will switch back to a secondary mode. In the secondary mode, it will sync up its own sync pulse to the primary sync pulse. The sync pulse receiver will have less tolerance in its sync pulse filtering mechanism than the secondary sync pulse transmitter. The sync pulse receiver will switch over more quickly than the secondary sync pulse transmitter. This is done to ensure that all receiver synchronizers will have switched over to using the secondary sync pulse transmitter source before the secondary sync pulse transmitter switches over to a primary mode.

Figure 5 shows sync pulse distribution.

In order to lockdown the backplane transmission from a fabric by the number of clock cycles indicated in the sync calculation, the entire fabric must effectively freeze for that many clock cycles to ensure that the same enqueuing and dequeueing decisions stay in sync. This requires support in each of the fabric ASICs. Lockdown stops all functionality, including special functions like queue resynch.

The sync signal from the synch pulse receiver is distributed to all ASICs. Each fabric ASIC contains a counter in

the core clock domain that counts clock cycles between global sync pulses. After the sync pulse if received, each ASIC calculates the number of clock cycles it is fast. (δ). Because the global sync is not transferred with its own clock, the calculated lockdown cycle value may not be the same for all ASICs on the same fabric. This difference is accounted for by keeping all interface FIFOs at a depth where they can tolerate the maximum skew of lockdown counts.

Lockdown cycles on all chips are always inserted at the same logical point relative to the beginning of the last sequence of "useful" (non-lockdown) cycles. That is, every chip will always execute the same number of "useful" cycles between lockdown events, even though the number of lockdown cycles varies.

Lockdown may occur at different times on different chips. All fabric input FIFOs are initially set up such that lockdown can occur on either side of the FIFO first without the FIFO running dry or overflowing. On each chip-chip interface, there is a sync FIFO to account for lockdown cycles (as well as board trace lengths and clock skews). The transmitter signals lockdown while it is locked down. The receiver does not push during indicated cycles, and does not pop during its own lockdown. The FIFO depth will vary depending on which chip locks down first, but the variation is bounded by the maximum number of lockdown cycles. The number of lockdown cycles a particular chip sees during one global sync period may vary, but they will all have the same number of useful cycles. The total number of lockdown cycles each chip on a particular fabric sees will be the same, within a bounded tolerance.

The Aggregator core clock domain completely stops for the lockdown duration - all flops and memory hold their state. Input FIFOs are allowed to build up. Lockdown bus cycles are inserted in the output queues. Exactly when the core lockdown is executed is dictated by when DOUT_AG bus protocol allows lockdown cycles to be inserted. DOUT_AG lockdown cycles are indicated on the DestID bus.

The memory controller must lockdown all flops for the appropriate number of cycles. To reduce impact to the silicon area in the memory controller, a technique called propagated lockdown is used.

The on-fabric chip-to-chip synchronization is executed at every sync pulse. While some sync error detecting capability may exist in some of the ASICs, it is the Unstriper's job to detect fabric synchronization errors and to remove the offending fabric. The chip-to-chip synchronization is a cascaded function that is done before any packet flow is enabled on the fabric. The synchronization flows from the Aggregator to the Memory Controller, to the Separator, and back to the Memory Controller. After the system reset, the Aggregators wait for the first global sync signal. When received, each Aggregator transmits a local sync command (value 0x2) on the DestID bus to each Memory Controller.

The striping function assigns bits from incoming data streams to individual fabrics. Two items were optimized in deriving the striping assignment:

1. Backplane efficiency should be optimized for OC48 and OC192

2. Backplane interconnection should not be significantly altered for OC192 operation.

These were traded off against additional muxing legs for the striper and unstriper ASICs. Irregardless of the optimization, the switch must have the same data format in the memory controller for both OC48 and OC192.

Backplane efficiency requires that minimal padding be added when forming the backplane busses. Given the 12 bit backplane bus for OC48 and the 48 bit backplane bus for OC192, an optimal assignment requires that the number of unused bits for a transfer to be equal to (number_of_bytes *8)/bus_width where "/" is integer division. For OC48, the bus can have 0, 4 or 8 unutilized bits. For OC192 the bus can have 0, 8, 16, 24, 32, or 40 unutilized bits.

This means that no bit can shift between 12 bit boundaries or else OC48 padding will not be optimal for certain packet lengths.

For OC192c, maximum bandwidth utilization means that each striper must receive the same number of bits (which implies bit interleaving into the stripers). When combined with the same backplane interconnection, this implies that in OC192c, each stripe must have exactly the correct number of bits come from each striper which has 1/4 of the bits.

For the purpose of assigning data bits to fabrics, a 48 bit frame is used. Inside the striper is a FIFO which is written 32 bits wide at 80-100 MHz and read 24 bits wide at 125 MHz. Three 32

bit words will yield four 24 bit words. Each pair of 24 bit words is treated as a 48 bit frame. The assignments between bits and fabrics depends on the number of fabrics.

TABLE 12: Bit striping function

		Fab 0	Fab 1	Fab 2	Fab 3	Fab 4	Fab 5	Fab 6	Fab 7	Fab 8	Fab 9	Fab 10	Fab 11
	0:11	0:11											
1 fab	12:23	12:23											
	24:35	24:35											
	36:47	36:47											
	0:11	0,2,5,7,8,10	1,3,4,6,9,11										
2 fab	12:23	13,15,16,18,21	12,14,17,19,20,22										
	24:35	+24 to 0:11	+24 to 0:11										
	36:47	+24 to 12:23	+24 to 12:23										
	0:11	0,3,5, 10	2,4,7, 9	1,6,8, 11									
3 fab	12:23	15,17,22,13	14,16,19,21	13,18,20,23									
	24:35	+24 to 0:11	+24 to 0:11	+24 to 0:11									
	36:47	+24 to 12:23	+24 to 12:23	+24 to 12:23									
	0:11	0,5,10	3,4,9	2,7,8	1,6,11								
4 fab	12:23	15,16,21	14,19,20	13,18,23	12,17,22								
	24:35	26,31,32	25,30,35	24,29,34	27,28,33								
	36:47	37,42,47	36,41,46	39,40,43	38,43,44								
	0:11	0,11	1,4	5,8	2,9	3,6	7,10						
6 fab	12:23	14,21	15,18	19,22	12,23	13,16	17,20						
	24:35	+24 to 0:11											
	36:47	+24 to 12:23											
	0:11	0	4	8	1	5	9	2	6	10	3	7	11
12 fab	12:23	15	19	23	12	16	20	13	17	21	14	18	22
	24:35	26	30	34	27	31	35	24	28	32	25	29	33
	36:47	37	41	45	38	42	46	39	43	47	37	40	44

The following tables give the byte lanes which are read first in the aggregator and written to first in the separator. The four channels are notated A,B,C,D. The different fabrics have different read/write order of the channels to allow for all busses to be fully utilized.

One fabric-40G

The next table gives the interface read order for the aggregator.

Fabric	1st	2nd	3rd	4th
0	A	B	C	D
Par	A	B	C	D

Two fabric-80G

Fabric	1st	2nd	3rd	4th
0	A	C	B	D
1	B	D	A	C
Par	A	C	B	D

120G

Fabric	1st	2nd	3rd	4th
0	A	D	B	C
1	C	A	D	B
2	B	C	A	D
Par	A	D	B	C

Three fabric-160G

Fabric	1st	2nd	3rd	4th
0	A	B	C	D
1	D	A	B	C
2	C	D	A	B
3	B	C	D	A
Par	A	B	C	D

Siz fabric-240 G

Fabric	1st	2nd	3rd	4th
0	A	D	C	B
1	B	A	D	C
2	B	A	D	C
3	C	B	A	D
4	D	C	B	A
5	D	C	B	A
Par	A	C	D	B

Twelve Fabric-480 G

Fabric	1st	2nd	3rd	4th
0,1,2	A	D	C	B
3,4,5	B	A	D	C
6,7,8	C	B	A	D
9,10,11	D	C	B	A
Par	A	B	C	D

Interfaces to the gigabit transceivers will utilize the transceiver bus as a split bus with two separate routeword and data busses. The routeword bus will be a fixed size (2 bits for OC48 ingress, 4 bits for OC48 egress, 8 bits for OC192 ingress and 16 bits for OC192 egress), the data bus is a variable sized bus. The transmit order will always have routeword bits at fixed locations. Every striping configuration has one transceiver that it used to

talk to a destination in all valid configurations. That transceiver will be used to send both routeword busses and to start sending the data.

The backplane interface is physically implemented using interfaces to the backplane transceivers. The bus for both ingress and egress is viewed as being composed of two halves, each with routeword data. The two bus halves may have information on separate packets if the first bus half ends a packet.

For example, an OC48 interface going to the fabrics locally speaking has 24 data bits and 2 routeword bits. This bus will be utilized acting as if it has 2x (12 bit data bus + 1 bit routeword bus). The two bus halves are referred to as A and B. Bus A is the first data, followed by bus B. A packet can start on either bus A or B and end on either bus A or B.

In mapping data bits and routeword bits to transceiver bits, the bus bits are interleaved. This ensures that all transceivers should have the same valid/invalid status, even if the striping amount changes. Routewords should be interpreted with bus A appearing before bus B.

The bus A/Bus B concept closely corresponds to having interfaces between chips.

All backplane busses support fragmentation of data. The protocol used marks the last transfer (via the final segment bit in the routeword). All transfers which are not final segment need to utilize the entire bus width, even if that is not an even number of

bytes. Any given packet must be striped to the same number of fabrics for all transfers of that packet. If the striping amount is updated in the striper during transmission of a packet, it will only update the striping at the beginning of the next packet.

Each transmitter on the ASICs will have the following I/O for each channel:

8 bit data bus, 1 bit clock, 1 bit control.

On the receive side, for channel the ASIC receives

a receive clock, 8 bit data bus, 3 bit status bus.

The switch optimizes the transceivers by mapping a transmitter to between 1 and 3 backplane pairs and each receiver with between 1 and 3 backplane pairs. This allows only enough transmitters to support traffic needed in a configuration to be populated on the board while maintaining a complete set of backplane nets. The motivation for this optimization was to reduce the number of transceivers needed.

The optimization was done while still requiring that at any time, two different striping amounts must be supported in the gigabit transceivers. This allows traffic to be enqueued from a striping data to one fabric and a striper striping data to two fabrics at the same time.

Depending on the bus configuration, multiple channels may need to be concatenated together to form one larger bandwidth pipe

(any time there is more than one transceiver in a logical connection. Although quad gbit transceivers can tie 4 channels together, this functionality is not used. Instead the receiving ASIC is responsible for synchronizing between the channels from one source. This is done in the same context as the generic synchronization algorithm.

The 8b/10b encoding/decoding in the gigabit transceivers allow a number of control events to be sent over the channel. The notation for these control events are K characters and they are numbered based on the encoded 10 bit value. Several of these K characters are used in the chipset. The K characters used and their functions are given in the table below.

TABLE 11: K Character usage

K character	Function	Notes
28.0	Sync indication	Transmitted after lockdown cycles, treated as the prime synchronization event at the receivers
28.1	Lockdown	Transmitted during lockdown cycles on the backplane
28.2	Packet Abort	Transmitted to indicate the card is unable to finish the current packet. Current use is limited to a port card being pulled while transmitting traffic
28.3'	Resync window	Transmitted by the striper at the start of a synch window if a resynch will be contained in the current sync window
28.4	BP set	Transmitted by the striper if the bus is currently idle and the value of the bp bit must be set.
28.5	Idle	Indicates idle condition
28.6	BP clr	Transmitted by the striper if the bus is currently idle and the bp bit must be cleared.

The switch has a variable number of data bits supported to each backplane channel depending on the striping configuration for a packet. Within a set of transceivers, data is filled in the following order:

F[fabric]_[oc192 port number][oc48 port designation (a,b,c,d)][transceiver_number]

The chipset implements certain functions which are described here. Most of the functions mentioned here have support in multiple ASICs, so documenting them on an ASIC by ASIC basis does not give a clear understanding of the full scope of the functions required.

The switch chipset is architected to work with packets up to 64K + 6 bytes long. On the ingress side of the switch, there are busses which are shared between multiple ports. For most packets, they are transmitted without any break from the start of packet to end of packet. However, this approach can lead to large delay variations for delay sensitive traffic. To allow delay sensitive traffic and long traffic to coexist on the same switch fabric, the concept of long packets is introduced. Basically long packets allow chunks of data to be sent to the queueing location, built up at the queueing location on a source basis and then added into the queue all at once when the end of the long packet is transferred. The definition of a long packet is based on the number of bits on each fabric.

If the switch is running in an environment where Ethernet MTU is maintained throughout the network, long packets will not be seen in a switch greater than 40G in size.

A wide cache-line shared memory technique is used to store cells/packets in the port/priority queues. The shared memory

stores cells/packets continuously so that there is virtually no fragmentation and bandwidth waste in the shared memory.

There exists up to 200 multiple queues in the shared memory. They are per-destination and priority based. All cells/packets which have the same output priority and blade/channel ID are stored in the same queue. Cells are always dequeued from the head of the list and enqueued into the tail of the queue. Each cell/packet consists of a portion of the egress route word, a packet length, and variable-length packet data. Cell and packets are stored continuously, i.e., the memory controller itself does not recognize the boundaries of cells/packets for the unicast connections. The packet length is stored for MC packets.

The multicast port mask memory 64Kx16-bit is used to store the destination port mask for the multicast connections, one entry (or multiple entries) per multicast VC. The port masks of the head multicast connections indicated by the multicast DestID FIFOs are stored internally for the scheduling reference. The port mask memory is retrieved when the port mask of head connection is cleaned and a new head connection is provided.

APS stands for a Automatic Protection Switching, which is a SONET redundancy standard. To support APS feature in the switch, two output ports on two different port cards send roughly the same traffic. The memory controllers maintain one set of queues for an APS port and send duplicate data to both output ports.

To support data duplication in the memory controller ASIC, each one of multiple unicast queues has a programmable APS

bit. If the APS bit is set to one, a packet is dequeued to both output ports. If the APS bit is set to zero for a port, the unicast queue operates at the normal mode. If a port is configured as an APS slave, then it will read from the queues of the APS master port. For OC48 ports, the APS port is always on the same OC48 port on the adjacent port card.

The shared memory queues in the memory controllers among the fabrics might be out of sync (i.e., same queues among different memory controller ASICs have different depths) due to clock drifts or a newly inserted fabric. It is important to bring the fabric queues to the valid and sync states from any arbitrary states. It is also desirable not to drop cells for any recovery mechanism.

A resync cell is broadcast to all fabrics (new and existing) to enter the resync state. Fabrics will attempt to drain all of the traffic received before the resynch cell before queue resynch ends, but no traffic received after the resynch cell is drained until queue resynch ends. A queue resynch ends when one of two events happens:

1. A timer expires.
2. The amount of new traffic (traffic received after the resynch cell) exceeds a threshold.

At the end of queue resynch, all memory controllers will flush any left-over old traffic (traffic received before the queue resynch cell). The freeing operation is fast enough to guarantee

that all memory controllers can fill all of memory no matter when the resynch state was entered.

Queue resynch impacts all 3 fabric ASICs. The aggregators must ensure that the FIFOs drain identically after a queue resynch cell. The memory controllers implement the queueing and dropping. The separators need to handle memory controllers dropping traffic and resetting the length parsing state machines when this happens. For details on support of queue resynch in individual ASICs, refer to the chip ADSs.

For the dequeue side, multicast connections have independent 32 tokens per port, each worth up 50-bit data or a complete packet. The head connection and its port mask of a higher priority queue is read out from the connection FIFO and the port mask memory every cycle. A complete packet is isolated from the multicast cache line based on the length field of the head connection. The head packet is sent to all its destination ports. The 8 queue drainers transmit the packet to the separators when there are non-zero multicast tokens are available for the ports. Next head connection will be processed only when the current head packet is sent out to all its ports.

Queue structure can be changed on fly through the fabric resync cell where the number of priority per port field is used to indicate how many priority queues each port has.

The following words have reasonably specific meanings in the vocabulary of the switch. Many are mentioned elsewhere, but this is an attempt to bring them together in one place with definitions.

TABLE 22:

Word	Meaning
APS	Automatic Protection Switching. A sonet/sdh standard for implementing redundancy on physical links. For the switch, APS is used to also recover from any detected port card failures.
Backplane synch	A generic term referring either to the general process the the switch boards use to account for varying transport delays between boards and clock drift or to the logic which implements the TX/RX functionality required for the the switch ASICs to account for varying transport delays and clock drifts.
BIB	The switch input bus. The bus which is used to pass data to the striper(s). See also BOB
Blade	Another term used for a port card. References to blades should have been eliminated from this document, but some may persist.
BOB	The switch output bus. The output bus from the striper which connects to the egress memory controller. See also BIB.
Egress Routeword	This is the routeword which is supplied to the chip after the unstriper. From an internal chipset perspective, the egress routeword is treated as data. See also fabric routeword.
Fabric Routeword	Routeword used by the fabric to determine the output queue. This routeword is not passed outside the unstriper. A significant portion of this routeword is blown away in the fabrics.
Freeze	Having logic maintain its values during lock-down cycles.
Lock-down	Period of time where the fabric effectively stops performing any work to compensate for clock drift. If the backplane synchronization logic determines that a fabric is 8 clock cycles fast, the fabric will lock down for 8 clocks.
Queue Resynch	A queue resynch is a series of steps executed to ensure that the logical state of all fabric queues for all ports is identical at one logical point in time. Queue resynch is not tied to backplane resynch (including lock- down) in any fashion, except that a lock-down can occur during a queue resynch.
SIB	Striped input bus. A largely obsolete term used to describe the output bus from the striper and input bus to the aggregator.
SOB	One of two meanings. The first is striped output bus, which is the output bus of the fabric and the input bus of the agg. See also SIB. The second meaning is a generic term used to describe engineers who left Marconi to form/work for a start-up after starting the switch design.
Sync	Depends heavily on context. Related terms are queue resynch, lock-down, freeze, and backplane sync.
Wacking	The implicit bit steering which occurs in the OC192 ingress stage since data is bit interleaved among stripers. This bit steering is reversed by the aggregators.

Although the invention has been described in detail in the foregoing embodiments for the purpose of illustration, it is to be understood that such detail is solely for that purpose and that

variations can be made therein by those skilled in the art without departing from the spirit and scope of the invention except as it may be described by the following claims.

APPENDIX

BFS Backplane optimization worksheet

Notes:

1) Colors are used to represent the use/reuse of serial communication channels.

idle
committed
used in next config (off limits for this config)
2nd use of receiver/transmitter
3rd use of receiver/transmitter

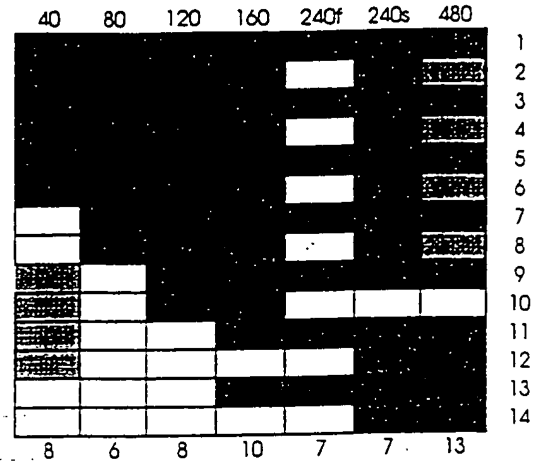
2) Channel assignments are optimized for multiple variables:

- Minimization of I/O. Quad GE transceivers provide 1:2 fanout buffers and 2:1multiplexers; 1:3 fanout or 3:1fanin requires additional external devices.

- Partitioning of PCB routing complexity. The ASIC to quad GE buses are ideally straight and interleaved (striper with unstriper, and aggregator with separator) to simplify the routing of these clocked buses. The serial connections from quad GE transceivers to card edge connector on fabric and blade require significant untangling (with minimal use of vias) to group signals by common destination on the edge connector. This is required to reduce the complexity of the backplane routing and corresponding layer count.

- Maximization of symmetry between ASIC's. This symmetry allows interleaving of buses anc reduces the need for external mux's and fanout buffers. It may also reduce the complexity of ASICs by reducing the number of unique mux/demux structures. Consequently, the striper map is a subset of the unstriper map, and both aggregator maps and the separator 5-8 map are derived from the separator 1-4 map.

	40	80	120	160	240f	240s	480	
Striper								
# active tx	8	6	8	10	7	7	13	
	10		10		7		[illegible]	
	[illegible]	8		12		13		
# fabrics	1	2	3	4	6	6	12	
configuration	40	80	120	160	240f	240s	480	
S_1A1	1	1	1	1	1	1	1	1
S_1A2	2	2	2	2				2
S_1A3	11							11
S_1A4	9							9
F1_1A1	3	3	3	3	3	3	3	3
F1_1A2	4	4	4	4				4
F1_1A3	12							12
F1_1A4	10							10
F2_1A1		5	5	5	5	5	5	5
F2_1A2		6	6	6				6
F3_1A1			7	7	7	7	7	7
F3_1A2			8	8				8
F4_1A1				9	9	9	9	9
F4_1A2				10				10
F5_1A1					11	11	11	11
F6_1A1					13	13	13	13
F7_1A1							2	2
F8_1A1							4	4
F9_1A1							6	6
F10_1A1							8	8
F11_1A1							12	12
F12_1A1							14	14
							Total	22



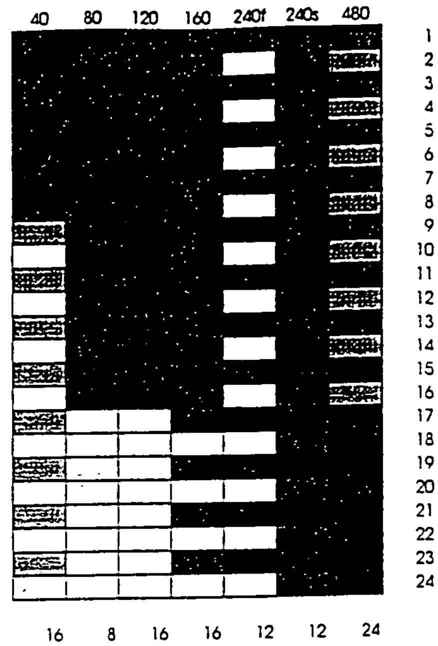
8 internal 1:2 fanout buffers
6 internal 1:1 fanout buffers (one output can be disab

Aggregator 1-4

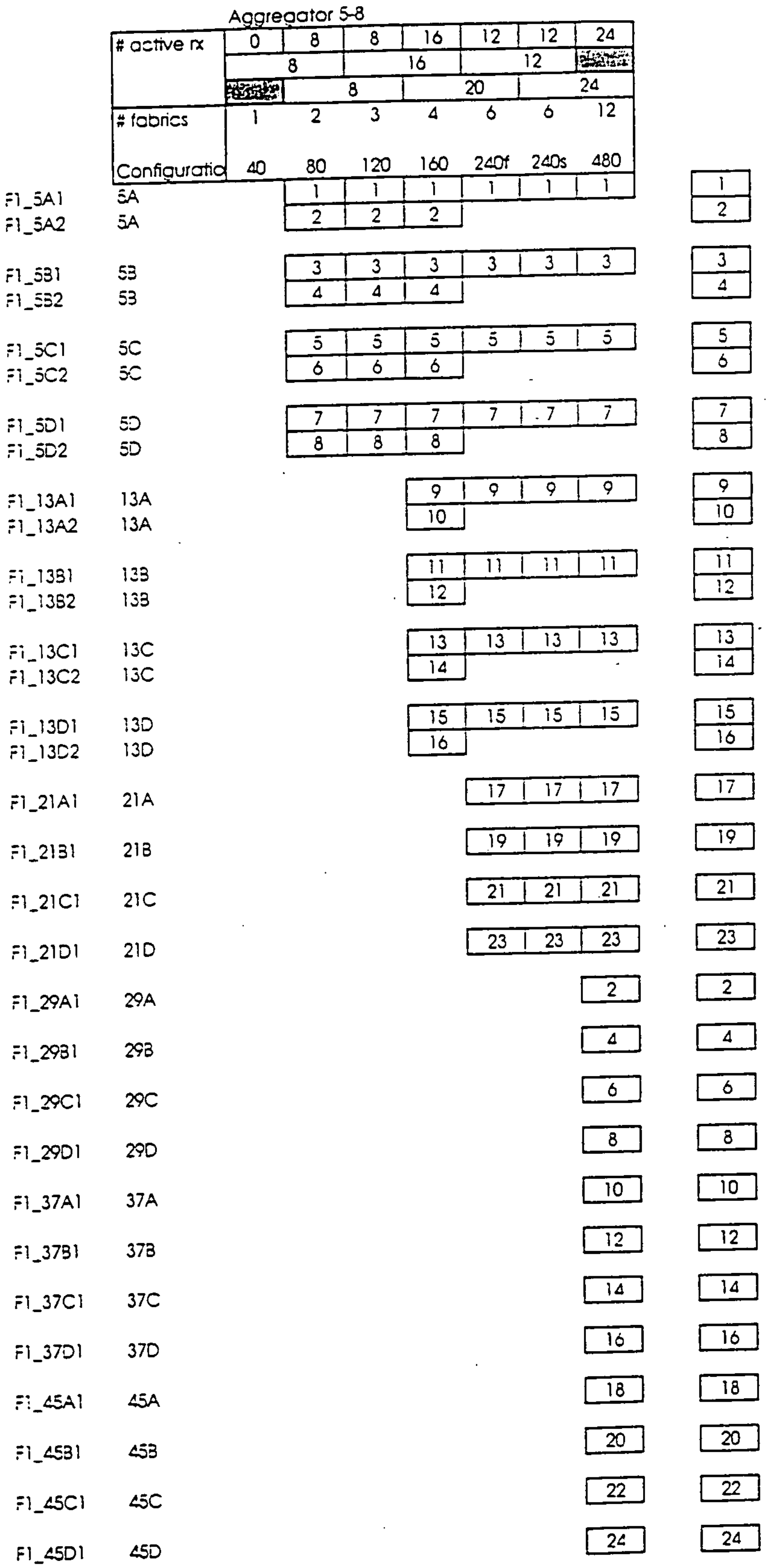
		40	80	120	160	240f	240s	480	
	# active rx	16	8	16	16	12	12	24	
			16		16		12		
				16		20		24	
	# fabrics	1	2	3	4	6	6	12	
	Configuratio	40	80	120	160	240f	240s	480	
F1_1A1	1A	1	1	1	1	1	1	1	1
F1_1A2	1A	2	2	2	2				2
F1_1A3	1A	17							17
F1_1A4	1A	9							9
F1_1B1	1B	3	3	3	3	3	3	3	3
F1_1B2	1B	4	4	4	4				4
F1_1B3	1B	19							19
F1_1B4	1B	11							11
F1_1C1	1C	5	5	5	5	5	5	5	5
F1_1C2	1C	6	6	6	6				6
F1_1C3	1C	21							21
F1_1C4	1C	13							13
F1_1D1	1D	7	7	7	7	7	7	7	7
F1_1D2	1D	8	8	8	8				8
F1_1D3	1D	23							23
F1_1D4	1D	15							15
F1_9A1	9A			9	9	9	9	9	9
F1_9A2	9A			10	10				10
F1_9B1	9B			11	11	11	11	11	11
F1_9B2	9B			12	12				12
F1_9C1	9C			13	13	13	13	13	13
F1_9C2	9C			14	14				14
F1_9D1	9D			15	15	15	15	15	15
F1_9D2	9D			16	16				16
F1_17A1	17A					17	17	17	17
F1_17B1	17B					19	19	19	19
F1_17C1	17C					21	21	21	21
F1_17D1	17D					23	23	23	23
F1_25A1	25A							2	2
F1_25B1	25B							4	4
F1_25C1	25C							6	6
F1_25D1	25D							8	8
F1_33A1	33A							10	10
F1_33B1	33B							12	12
F1_33C1	33C							14	14
F1_33D1	33D							16	16
F1_41A1	41A							18	18
F1_41B1	41B							20	20
F1_41C1	41C							22	22
F1_41D1	41D							24	24

increments by 8

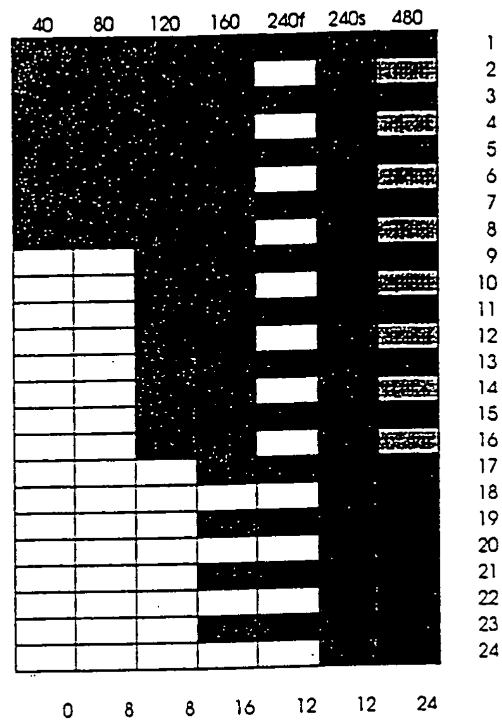
40 total



16 internal 2:1 Mux's
8 internal 1:1 buffers (one input can be turned off)



32 total



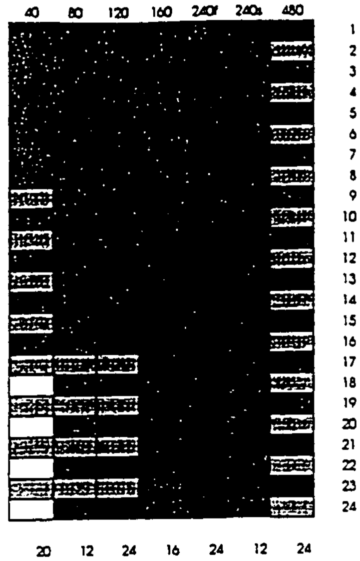
8 internal 2:1 Mux's
16 internal 1:1 buffers (one input can be turned off)

Separator 1-4

	# active tx	20	12	24	16	24	12	24	
		20		24		24			
			24		24		24		
	# taoncs	1	2	3	4	6	6	12	
	Configuratio	40	80	120	160	240f	240s	480	
F1_1A1	1A	1	1	1	1	1	1	1	1
F1_1A2	1A	2	2	2	2	2			2
F1_1A3	1A	17	17	17					17
F1_1A4	1A	9							9
F1_1A5	1A	10							10
F1_1B1	1B	3	3	3	3	3	3	3	3
F1_1B2	1B	4	4	4	4	4			4
F1_1B3	1B	19	19	19					19
F1_1B4	1B	11							11
F1_1B5	1B	12							12
F1_1C1	1C	5	5	5	5	5	5	5	5
F1_1C2	1C	6	6	6	6	6			6
F1_1C3	1C	21	21	21					21
F1_1C4	1C	13							13
F1_1C5	1C	14							14
F1_1D1	1D	7	7	7	7	7	7	7	7
F1_1D2	1D	8	8	8	8	8			8
F1_1D3	1D	23	23	23					23
F1_1D4	1D	15							15
F1_1D5	1D	16							16
F1_9A1	9A			9	9	9	9	9	9
F1_9A2	9A			10	10	10			10
F1_9A3	9A			18					18
F1_9B1	9B			11	11	11	11	11	11
F1_9B2	9B			12	12	12			12
F1_9B3	9B			20					20
F1_9C1	9C			13	13	13	13	13	13
F1_9C2	9C			14	14	14			14
F1_9C3	9C			22					22
F1_9D1	9D			15	15	15	15	15	15
F1_9D2	9D			16	16	16			16
F1_9D3	9D			24					24
F1_17A1	17A					17	17	17	17
F1_17A2	17A					18			18
F1_17B1	17B					19	19	19	19
F1_17B2	17B					20			20
F1_17C1	17C					21	21	21	21
F1_17C2	17C					22			22
F1_17D1	17D					23	23	23	23
F1_17D2	17D					24			24
F1_25A1	25A							2	2
F1_25B1	25B							4	4
F1_25C1	25C							6	6
F1_25D1	25D							8	8
F1_33A1	33A							10	10
F1_33B1	33B							12	12
F1_33C1	33C							14	14
F1_33D1	33D							16	16
F1_41A1	41A							18	18
F1_41B1	41B							20	20
F1_41C1	41C							22	22
F1_41D1	41D							24	24

increments by 8

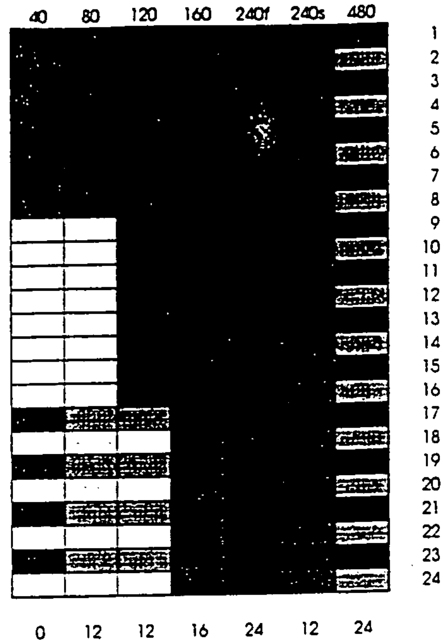
52 total



8 external 1:2 fanout buffers
20 internal 1:2 fanout buffers
4 internal 1:1 buffers (one output can be turned off)

Separator 5-8

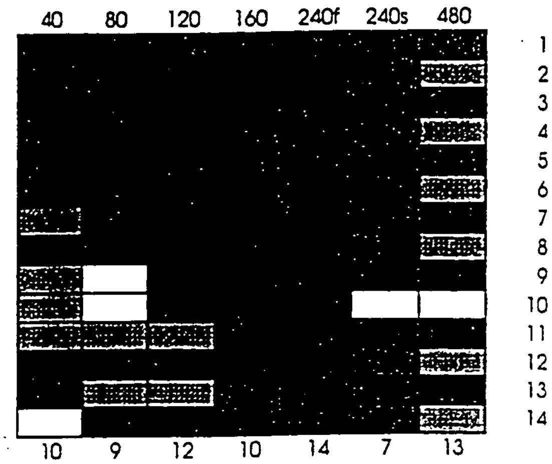
		40	80	120	160	240f	240s	480	
# active tx		0	12	12	16	24	12	24	
		12		20		24		[shaded]	
		[shaded]	12		24		24		
# fabrics		1	2	3	4	6	6	12	
Configuratio		40	80	120	160	240f	240s	480	
F1_5A1	5A		1	1	1	1	1	1	1
F1_5A2	5A		2	2	2	2			2
F1_5A3	5A		17	17					17
F1_5B1	5B		3	3	3	3	3	3	3
F1_5B2	5B		4	4	4	4			4
F1_5B3	5B		19	19					19
F1_5C1	5C		5	5	5	5	5	5	5
F1_5C2	5C		6	6	6	6			6
F1_5C3	5C		21	21					21
F1_5D1	5D		7	7	7	7	7	7	7
F1_5D2	5D		8	8	8	8			8
F1_5D3	5D		23	23					23
F1_13A1	13A				9	9	9	9	9
F1_13A2	13A				10	10			10
F1_13B1	13B				11	11	11	11	11
F1_13B2	13B				12	12			12
F1_13C1	13C				13	13	13	13	13
F1_13C2	13C				14	14			14
F1_13D1	13D				15	15	15	15	15
F1_13D2	13D				16	16			16
F1_21A1	21A					17	17	17	17
F1_21A2	21A					18			18
F1_21B1	21B					19	19	19	19
F1_21B2	21B					20			20
F1_21C1	21C					21	21	21	21
F1_21C2	21C					22			22
F1_21D1	21D					23	23	23	23
F1_21D2	21D					24			24
F1_29A1	29A							2	2
F1_29B1	29B							4	4
F1_29C1	29C							6	6
F1_29D1	29D							8	8
F1_37A1	37A							10	10
F1_37B1	37B							12	12
F1_37C1	37C							14	14
F1_37D1	37D							16	16
F1_45A1	45A							18	18
F1_45B1	45B							20	20
F1_45C1	45C							22	22
F1_45D1	45D							24	24
									40 total



16 internal 1:2 fanout buffers
12 internal 1:1 buffers (one output can be turned off)

Unstriper

# active rx	10	9	12	10	14	7	13	
	13		14		14			
		12		14		13		
# fabrics	1	2	3	4	6	6	12	
Configuration	40	80	120	160	240f	240s	480	
S_1A1	1	1	1	1	1	1	1	1
S_1A2	2	2	2	2	2			2
S_1A3	11	11	11					11
S_1A4	9							9
S_1A5	7							7
F1_1A1	3	3	3	3	3	3	3	3
F1_1A2	4	4	4	4	4			4
F1_1A3	12	12	12					12
F1_1A4	10							10
F1_1A5	8							8
F2_1A1		5	5	5	5	5	5	5
F2_1A2		6	6	6	6			6
F2_1A3		13	13					13
F3_1A1			7	7	7	7	7	7
F3_1A2			8	8	8			8
F3_1A3			14					14
F4_1A1				9	9	9	9	9
F4_1A2				10	10			10
F5_1A1					11	11	11	11
F5_1A2					12			12
F6_1A1					13	13	13	13
F6_1A2					14			14
F7_1A1							2	2
F8_1A1							4	4
F9_1A1							6	6
F10_1A1							8	8
F11_1A1							12	12
F12_1A1							14	14
							Total	28



3 external 2:1 Mux's
11 internal 2:1 Mux's
3 internal 1:1 input buffers (one input can be disabled

BFS ASIC Backplane Connection Map

Revision	Date	Comments
1	11/3/99	First release, checked by Speiser
2	11/8/99	Changed 40G assignments for channels 7 and 9; affects the striper and unstriper only. Changed 40, 80, 120G assignments for channels 17-24 to make Separator 1-4 and Separator 5-8 symmetric (Separator 5-8 is now a subset of Separator 1-4) Checked by Speiser and Blanchini
3	11/30/99	Added control port assignments.

Naming convention:

Fabric to blade egress bus format: <fabric>_<blade><channel><lane><polarity>				
Fabric slot	Blade slot	Channel	Lane	Polarity
F1-F12,S	1-48	A-D	1-5	P,N

Notes:

1) Map shown for a single instance of each ASIC, e.g. striper channel A for blade number 1.
From an ASIC point of view, all that is significant in this striper table is the fabric number and the lane number; the topology is the same for each subsequent channel (B-D) blade slot (2-48).

2) Polarity is required to distinguish the two physical wires forming a differential pair.
For each table entry (i.e., each logical signal) there are two physical connections.
For example, logical signal F2_1A3 is really comprised of F2_1A3P and F2_1A3N, which connect fabric 2 to blade 1, unstriper 1, lane 3. Polarity is not shown in the connection maps to make them easier to read.

3) How to use this map: For a given configuration (e.g. with one fabric installed, read down 40 G column)
Read down the column to determine the source for data present on an striper output bus.
Each row represents a single striper output bus.

4) = not used in this Configuration

5) To aid in PCB routing and reduce design complexity, there are only two unique connection assignments, the unstriper and the Separator 1-4 maps. The striper map is a subset of the unstriper map.
The separator 5-8 map, the aggregator 1-4 map, and the aggregator 5-8 map are all subsets of the separator 1-4 map.

6) Up to four control ports are supported (see maps for aggregator 9 and separator 9). If not all four control ports are needed, some control port channels can be deleted. For example, if only two 2.5Gbps control ports are needed, control port channels C and D can be deleted.

Striper

Configuration							Striper Output bus
40	80	120	160	240f	240s	480	
S_1A1	S_1A1	S_1A1	S_1A1	S_1A1	S_1A1	S_1A1	1
S_1A2	S_1A2	S_1A2	S_1A2			F7_1A1	2
F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	3
F1_1A2	F1_1A2	F1_1A2	F1_1A2			F8_1A1	4
	F2_1A1	F2_1A1	F2_1A1	F2_1A1	F2_1A1	F2_1A1	5
	F2_1A2	F2_1A2	F2_1A2			F9_1A1	6
		F3_1A1	F3_1A1	F3_1A1	F3_1A1	F3_1A1	7
		F3_1A2	F3_1A2			F10_1A1	8
S_1A4			F4_1A1	F4_1A1	F4_1A1	F4_1A1	9
F1_1A4			F4_1A2				10
S_1A3				F5_1A1	F5_1A1	F5_1A1	11
F1_1A3						F11_1A1	12
				F6_1A1	F6_1A1	F6_1A1	13
						F12_1A1	14

Aggregator 1-4

Configuration							Aggregator Input bus
40	80	120	160	240f	240s	480	
F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	1
F1_1A2	F1_1A2	F1_1A2	F1_1A2			F1_25A1	2
F1_1B1	F1_1B1	F1_1B1	F1_1B1	F1_1B1	F1_1B1	F1_1B1	3
F1_1B2	F1_1B2	F1_1B2	F1_1B2			F1_25B1	4
F1_1C1	F1_1C1	F1_1C1	F1_1C1	F1_1C1	F1_1C1	F1_1C1	5
F1_1C2	F1_1C2	F1_1C2	F1_1C2			F1_25C1	6
F1_1D1	F1_1D1	F1_1D1	F1_1D1	F1_1D1	F1_1D1	F1_1D1	7
F1_1D2	F1_1D2	F1_1D2	F1_1D2			F1_25D1	8
F1_1A4		F1_9A1	F1_9A1	F1_9A1	F1_9A1	F1_9A1	9
		F1_9A2	F1_9A2			F1_33A1	10
F1_1B4		F1_9B1	F1_9B1	F1_9B1	F1_9B1	F1_9B1	11
		F1_9B2	F1_9B2			F1_33B1	12
F1_1C4		F1_9C1	F1_9C1	F1_9C1	F1_9C1	F1_9C1	13
		F1_9C2	F1_9C2			F1_33C1	14
F1_1D4		F1_9D1	F1_9D1	F1_9D1	F1_9D1	F1_9D1	15
		F1_9D2	F1_9D2			F1_33D1	16
F1_1A3				F1_17A1	F1_17A1	F1_17A1	17
						F1_41A1	18
F1_1B3				F1_17B1	F1_17B1	F1_17B1	19
						F1_41B1	20
F1_1C3				F1_17C1	F1_17C1	F1_17C1	21
						F1_41C1	22
F1_1D3				F1_17D1	F1_17D1	F1_17D1	23
						F1_41D1	24

Aggregator 5-8

Configuration							Aggregator Input bus
40	80	120	160	240f	240s	480	
	F1_5A1	F1_5A1	F1_5A1	F1_5A1	F1_5A1	F1_5A1	1
	F1_5A2	F1_5A2	F1_5A2			F1_29A1	2
	F1_5B1	F1_5B1	F1_5B1	F1_5B1	F1_5B1	F1_5B1	3
	F1_5B2	F1_5B2	F1_5B2			F1_29B1	4
	F1_5C1	F1_5C1	F1_5C1	F1_5C1	F1_5C1	F1_5C1	5
	F1_5C2	F1_5C2	F1_5C2			F1_29C1	6
	F1_5D1	F1_5D1	F1_5D1	F1_5D1	F1_5D1	F1_5D1	7
	F1_5D2	F1_5D2	F1_5D2			F1_29D1	8
			F1_13A1	F1_13A1	F1_13A1	F1_13A1	9
			F1_13A2			F1_37A1	10
			F1_13B1	F1_13B1	F1_13B1	F1_13B1	11
			F1_13B2			F1_37B1	12
			F1_13C1	F1_13C1	F1_13C1	F1_13C1	13
			F1_13C2			F1_37C1	14
			F1_13D1	F1_13D1	F1_13D1	F1_13D1	15
			F1_13D2			F1_37D1	16
				F1_21A1	F1_21A1	F1_21A1	17
						F1_45A1	18
				F1_21B1	F1_21B1	F1_21B1	19
						F1_45B1	20
				F1_21C1	F1_21C1	F1_21C1	21
						F1_45C1	22
				F1_21D1	F1_21D1	F1_21D1	23
						F1_45D1	24

Aggregator 9

Configuration							Aggregator Input bus
40	80	120	160	240f	240s	480	
F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	1
F1_CP_A2	F1_CP_A2	F1_CP_A2	F1_CP_A2				2
F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	3
F1_CP_B2	F1_CP_B2	F1_CP_B2	F1_CP_B2				4
F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	5
F1_CP_C2	F1_CP_C2	F1_CP_C2	F1_CP_C2				6
F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	7
F1_CP_D2	F1_CP_D2	F1_CP_D2	F1_CP_D2				8
F1_CP_A4							9
							10
F1_CP_B4							11
							12
F1_CP_C4							13
							14
F1_CP_D4							15
							16
F1_CP_A3							17
							18
F1_CP_B3							19
							20
F1_CP_C3							21
							22
F1_CP_D3							23
							24

Separator 1-4

Configuration							Separator output bus
40	80	120	160	240f	240s	480	
F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	1
F1_1A2	F1_1A2	F1_1A2	F1_1A2	F1_1A2		F1_25A1	2
F1_1B1	F1_1B1	F1_1B1	F1_1B1	F1_1B1	F1_1B1	F1_1B1	3
F1_1B2	F1_1B2	F1_1B2	F1_1B2	F1_1B2		F1_25B1	4
F1_1C1	F1_1C1	F1_1C1	F1_1C1	F1_1C1	F1_1C1	F1_1C1	5
F1_1C2	F1_1C2	F1_1C2	F1_1C2	F1_1C2		F1_25C1	6
F1_1D1	F1_1D1	F1_1D1	F1_1D1	F1_1D1	F1_1D1	F1_1D1	7
F1_1D2	F1_1D2	F1_1D2	F1_1D2	F1_1D2		F1_25D1	8
F1_1A4		F1_9A1	F1_9A1	F1_9A1	F1_9A1	F1_9A1	9
F1_1A5		F1_9A2	F1_9A2	F1_9A2		F1_33A1	10
F1_1B4		F1_9B1	F1_9B1	F1_9B1	F1_9B1	F1_9B1	11
F1_1B5		F1_9B2	F1_9B2	F1_9B2		F1_33B1	12
F1_1C4		F1_9C1	F1_9C1	F1_9C1	F1_9C1	F1_9C1	13
F1_1C5		F1_9C2	F1_9C2	F1_9C2		F1_33C1	14
F1_1D4		F1_9D1	F1_9D1	F1_9D1	F1_9D1	F1_9D1	15
F1_1D5		F1_9D2	F1_9D2	F1_9D2		F1_33D1	16
F1_1A3	F1_1A3	F1_1A3		F1_17A1	F1_17A1	F1_17A1	17
		F1_9A3		F1_17A2		F1_41A1	18
F1_1B3	F1_1B3	F1_1B3		F1_17B1	F1_17B1	F1_17B1	19
		F1_9B3		F1_17B2		F1_41B1	20
F1_1C3	F1_1C3	F1_1C3		F1_17C1	F1_17C1	F1_17C1	21
		F1_9C3		F1_17C2		F1_41C1	22
F1_1D3	F1_1D3	F1_1D3		F1_17D1	F1_17D1	F1_17D1	23
		F1_9D3		F1_17D2		F1_41D1	24

Separator 5-8

Configuration							Separator output bus
40	80	120	160	240f	240s	480	
	F1_5A1	F1_5A1	F1_5A1	F1_5A1	F1_5A1	F1_5A1	1
	F1_5A2	F1_5A2	F1_5A2	F1_5A2		F1_29A1	2
	F1_5B1	F1_5B1	F1_5B1	F1_5B1	F1_5B1	F1_5B1	3
	F1_5B2	F1_5B2	F1_5B2	F1_5B2		F1_29B1	4
	F1_5C1	F1_5C1	F1_5C1	F1_5C1	F1_5C1	F1_5C1	5
	F1_5C2	F1_5C2	F1_5C2	F1_5C2		F1_29C1	6
	F1_5D1	F1_5D1	F1_5D1	F1_5D1	F1_5D1	F1_5D1	7
	F1_5D2	F1_5D2	F1_5D2	F1_5D2		F1_29D1	8
			F1_13A1	F1_13A1	F1_13A1	F1_13A1	9
			F1_13A2	F1_13A2		F1_37A1	10
			F1_13B1	F1_13B1	F1_13B1	F1_13B1	11
			F1_13B2	F1_13B2		F1_37B1	12
			F1_13C1	F1_13C1	F1_13C1	F1_13C1	13
			F1_13C2	F1_13C2		F1_37C1	14
			F1_13D1	F1_13D1	F1_13D1	F1_13D1	15
			F1_13D2	F1_13D2		F1_37D1	16
	F1_5A3	F1_5A3		F1_21A1	F1_21A1	F1_21A1	17
				F1_21A2		F1_45A1	18
	F1_5B3	F1_5B3		F1_21B1	F1_21B1	F1_21B1	19
				F1_21B2		F1_45B1	20
	F1_5C3	F1_5C3		F1_21C1	F1_21C1	F1_21C1	21
				F1_21C2		F1_45C1	22
	F1_5D3	F1_5D3		F1_21D1	F1_21D1	F1_21D1	23
				F1_21D2		F1_45D1	24

Separator 9

Configuration							Separator output bus
40	80	120	160	240f	240s	480	
F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	F1_CP_A1	1
F1_CP_A2	F1_CP_A2	F1_CP_A2	F1_CP_A2	F1_CP_A2			2
F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	F1_CP_B1	3
F1_CP_B2	F1_CP_B2	F1_CP_B2	F1_CP_B2	F1_CP_B2			4
F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	F1_CP_C1	5
F1_CP_C2	F1_CP_C2	F1_CP_C2	F1_CP_C2	F1_CP_C2			6
F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	F1_CP_D1	7
F1_CP_D2	F1_CP_D2	F1_CP_D2	F1_CP_D2	F1_CP_D2			8
F1_CP_A4							9
F1_CP_A5							10
F1_CP_B4							11
F1_CP_B5							12
F1_CP_C4							13
F1_CP_C5							14
F1_CP_D4							15
F1_CP_D5							16
F1_CP_A3	F1_CP_A3	F1_CP_A3					17
							18
F1_CP_B3	F1_CP_B3	F1_CP_B3					19
							20
F1_CP_C3	F1_CP_C3	F1_CP_C3					21
							22
F1_CP_D3	F1_CP_D3	F1_CP_D3					23
							24

Unstriper

Configuration							Unstriper Input bus
40	80	120	160	240f	240s	480	
S_1A1	S_1A1	S_1A1	S_1A1	S_1A1	S_1A1	S_1A1	1
S_1A2	S_1A2	S_1A2	S_1A2	S_1A2		F7_1A1	2
F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	F1_1A1	3
F1_1A2	F1_1A2	F1_1A2	F1_1A2	F1_1A2		F8_1A1	4
	F2_1A1	F2_1A1	F2_1A1	F2_1A1	F2_1A1	F2_1A1	5
	F2_1A2	F2_1A2	F2_1A2	F2_1A2		F9_1A1	6
S_1A5		F3_1A1	F3_1A1	F3_1A1	F3_1A1	F3_1A1	7
F1_1A5		F3_1A2	F3_1A2	F3_1A2		F10_1A1	8
S_1A4			F4_1A1	F4_1A1	F4_1A1	F4_1A1	9
F1_1A4			F4_1A2	F4_1A2			10
S_1A3	S_1A3	S_1A3		F5_1A1	F5_1A1	F5_1A1	11
F1_1A3	F1_1A3	F1_1A3		F5_1A2		F11_1A1	12
	F2_1A3	F2_1A3		F6_1A1	F6_1A1	F6_1A1	13
		F3_1A3		F6_1A2		F12_1A1	14

gigabit signals

Switch size		40	80	120	160	240f	240s	480	<-- gigabits	
number of fabrics		1	2	3	4	6	6	12	<-- does not include spare	
base data bus		12	6	4	3	2	2	1	<-- number of 250MHz data bits from striper to aggregator per OC-48 (4x per port card)	
Calculated										
striper to fabric bus	data in	26	14	10	8	6	6	4	@125MHz to Tx	
	gigabits	3.25	1.75	1.25	1	0.75	0.75	0.5	/8 bits per Tx	
	transmitters	4	2	2	1	1	1	1	round up to unit Tx	
fabric to unstriper bus	data out	52	28	20	16	12	12	8	@125MHz from Rx (w/2x speed-up)	
	gigabits	6.5	3.5	2.5	2	1.5	1.5	1	/8 bits per Rx	
	receivers	7	4	3	2	2	2	1	round up to unit Rx	
Implemented										
striper to fabric bus	transmitters	4	2	2	2	1	1	1	<-- update these numbers to affect input speedup	
fabric to unstriper bus	receivers	5	3	3	2	2	1	1	<-- update these numbers to affect output speedup	
	speedup	1.50	1.67	2.00	2.00	2.00	1.00	2.00	speed-up assumes always use 4 RW bits in egress direction	
Configs		40/80	80/120	120/160	160/240f	240f/240s	240s/480		required simultaneous modes	
Port Card										
striper	active Tx	10	8	10	12	7	13	13	max active Tx	number of serializers per striper
unstriper	active Rx	13	12	14	14	14	13	14	max active Rx	number of deserializers per unstriper
unique streams								56		number of trancelvers per port card
								$\boxed{14}$		number of quad trancelvers per port card
total streams										
striper	Tx increment	10	2	2	2	0	6	22	total Tx	number of gigabit backplane connections per striper
unstriper	Rx increment	13	3	2	4	0	6	28	total Rx	number of gigabit backplane connections per unstriper
								200		number of gigabit backplane connections per port card
								$\boxed{400}$		number of pins per port card
Fabric										
Agg1-4	active Rx	16	16	16	20	12	24	24	max active Rx	number of deserializers per odd aggregator
Agg5-8	active Rx	8	8	16	20	12	24	24	max active Rx	number of deserializers per even aggregator
Sep1-4	active Tx	20	24	24	24	24	24	24	max active Tx	number of serializers per odd separator
Sep5-8	active Tx	12	12	20	24	24	24	24	max active Tx	number of serializers per even separator
unique streams								192		number of trancelvers per fabric
								$\boxed{48}$		number of quad trancievers per fabric
total streams										
Agg1-4	Rx increment	16	8	0	4	0	12	40	total Rx	number of gigabit backplane connections per odd aggregator
Agg5-8	Rx increment	8	0	8	4	0	12	32	total Rx	number of gigabit backplane connections per even aggregator
Sep1-4	Tx increment	20	12	0	8	0	12	52	total Tx	number of gigabit backplane connections per odd separator
Sep5-8	Tx increment	12	0	8	8	0	12	40	total Tx	number of gigabit backplane connections per even separator
								656		number of gigabit backplane connections per fabric
								$\boxed{1312}$		number of pins per fabric